RFI #2



# Related Work

Earlier studies on the performance of CORBA objects focussed mainly on identifying the performance constraints of an Object Request Broker (ORB). Schmidt analyzed the performance of Orbix and VisiBroker over high speed ATM networks and pointed out key sources of overhead in middleware ORBs [9,10]. This thesis complements Schmidt's work by demonstrating an integrated and automated approach which is capable of separately measuring the influence of the ORB, the operating system and the underlying network on the performance of CORBA objects.

Studies have also been conducted on IDL compiler optimizations that can improve overall performance of the ORB. One such effort is the Flick project[6]. It is claimed that Flick-generated stubs marshal data between 2 and 17 times faster than stubs produced by traditional IDL compilers. This results in an increased end-to-end throughput by factors between 1.2 and 3.7.

A number of efforts have been made to measure the performance of ORBs. Sometimes ORB vendors publish the performance of their ORB and compare it with other ORBs [20]. Other approaches also exist which use ad-hoc techniques to compare the performance of different ORBs. These techniques generally measure only specific aspects of the performance of an ORB. It is important to realize that just a few simple tests *do not mean that the aggregate performance of ORB A is better than ORB B*. Sometimes, ORB vendors tend to optimize the performance of their ORB for a few types of tests. We need to consider the combined results from tests which bring out all aspects of ORB performance (such as scalability tests, marshaling tests, de-multiplexing tests, latency tests, etc.). Another important aspect about the performance of CORBA objects is that it is very sensitive to the operating environment (OS, Network, ORB middleware, compiler optimization, thread package etc.). Variations in operating environment can affect end-to-end performance of CORBA objects. Hence, it is better for end-users to run performance tests in the target operating environment rather than to depend on results obtained under other operating environments.

The real-time CORBA document created by RTSIG describes an architecture overview of a CORBA ORB for real-time applications [16]. It does not target any specific type of real-time application. Instead, it brings out general features that should be supported by a real-time ORB and provides pointers on how to implement these features. A real-time API is defined in a separate module called RT. The separation of RT extensions into a separate module is done so that conventional ORBs are not obliged to implement them. The document also discusses a real-time scheduling service (RTSS) which can maintain the notion of global priority. The IDL for the RTSS also describes functions that can be used to set and get priority. Although a priority-based approach can be used to provide QoS guarantees to many real-time systems, it may not be suited for isochronous real-time applications such as multimedia. Our RTSS provides access to both priority-based preemptive scheduling and explicit plan scheduling.

TAO is the ACE ORB being developed at Washington University [28]. This project focuses on: (1) identifying the enhancements required to standard ORB specifications that will enable applications to specify their Quality of Service (QoS) requirements to ORBs, (2) determining features required to build real-time ORBs, (3) integrating the strategies for I/O subsystem architectures and optimizations with ORB middleware, and (4) to capture and document key design patterns necessary to develop, maintain and extend real-time ORB middleware. In addition to providing a real-time ORB, TAO is an integrated ORB endsystem architecture that consists of a high-performance I/O subsystem and an ATM port inter-

connect controller (APIC).

Schmidt has pointed out some causes of performance overhead in middleware and ORB endsystems [9,10,29]. They include: (1) non-optimized presentation layer conversions, data copying, and memory management, (2) inefficient receiver-side demultiplexing, (3) excessive control information carried in request messages, (4) inefficient design of network adapters, (5) inefficient protocol implementations and improper integration with I/O subsystem, (6) inefficient implementations of ORB transport protocols, and (7) lack of proper integration with the operating system. The integrated approach described in this thesis can be used to create a comprehensive benchmark test suite that helps to identify and quantify the effects of the performance overhead mentioned above. Heterogeneous firm real-time systems often consist of ORBs, operating systems and networks from different vendors. Some vendors may focus on performance optimizations while others may focus more on the functional aspects. Hence it is important to determine the effects of the presence of performance constraints and to what extent they affect the end-to-end performance observed by real-time objects.

One of the commercially available CORBA test suites is the VSORB from X/Open [23]. VSORB is implemented under the TETware test harness, a version of the Test Environment Toolkit (TET), a widely used framework for implementing test suites [18]. It is designed for two primary uses (1) testing ORB implementations for CORBA conformance and interoperability under formal processes and procedures (2) CORBA compliance testing by ORB implementors during product development and quality assurance.

Wolfe *et al* at URI have developed a dynamic real-time CORBA system which supports the expression and enforcement of end-to-end timing constraints [36]. Clients can express timing constraints through Timed Distributed Method Invocations (TDMI) and the system enforces the timing constraints through various extensions and additions to the CORBA standard. A Real-Time Event Service enforces the distribution of real-time events in priority order with real-time enforcement of event response time. Their Global Priority Service is very similar to the Real-Time Scheduling Service presented in this thesis but differs in the support for both a priority- based preemptive scheduling and explicit plan scheduling schemes.

The QuO research at BBN identifies key issues that need to be addressed to support QoS at the CORBA object layer, especially across wide-area and mobile environments [19]. They have developed an architecture, Quality of Service for CORBA Objects (QuO), to support QoS at the CORBA layer. QuO extends the CORBA functional Interface Description Language (IDL) with a QoS Description Language (QDL). QDL specifies an application's expected usage patterns and QoS requirements for a connection to an object. Although the work in this thesis does not involve modifications to middleware to support variations in operating environment, its effect on the performance of CORBA objects can be easily studied.

A number of other real-time ORBs exist in the market today with varying levels of QoS support to real-time applications. Iona has ported its ORB to run on real-time systems such as VxWorks and QNX [32]. DIMMA, the microkernel ORB from ANSA, [22] and RCP-ORB from Nortel [24] are also quite promising. However a lot of work still needs to be done to provide end-to-end QoS support for real-time objects.

---

Next Up Previous Contents

**Next:** Implementation **Up:** No Title **Previous:** Introduction
*Sridhar Nimmagadda*

*9/14/1998*